Generic Platform for Manufacturing Execution System Functions in Knowledge-Driven Manufacturing Systems

Wael M. Mohammed[1], Borja Ramis Ferrer[1], Sergii Iarovyi[1], Elisa Negri[2], Luca Fumagalli[2], Andrei Lobov[1], and Jose L. Martinez Lastra[1]

[1] *FAST-Lab, Tampere University of Technology, Tampere, Finland*

P.O. 600, FI-33101 Tampere, Finland

[2] *Department of Management Engineering, Politecnico di Milano, Milan, Italy*

Piazza Leonardo da Vinci, 32, 20133 Milano - Italy

wael.mohammed@tut.fi, borja.ramisferrer@tut.fi, sergii.iarovyi@tut.fi, elisa.negri@polimi.it, luca1.fumagalli@polimi.it, andrei.lobov@tut.fi, jose.lastra@tut.fi

Generic Platform for Manufacturing Execution System Functions in Knowledge-Driven Manufacturing Systems

Information technologies grow rapidly nowadays with the advance and extension of computing capabilities. This growth affects several fields, which consume these technologies. Industrial Automation is not an exception. This publication describes a general and flexible architecture for implementing Manufacturing Execution System (MES) function, which can be deployed in multiple industrial case. These features are achieved by combining the flexibility of knowledge-driven systems with the vendor-independent property of RESTful web services. With deployment of this solution, MES functions may gain more versatility and independency. This research work is a continuation of the development of the OKD-MES (Open Knowledge-Driven Manufacturing Execution System) framework during the execution of the eScop project. The OKD-MES framework consists on a semantic-based solution for controlling and enhancing the flexibility and re-configurability of MES. In such scope, this research presents MES functions architecture that might be implemented in the OKD-MES framework in order to increase the flexibility of event-driven manufacturing systems.

Keywords: knowledge-driven manufacturing systems; manufacturing execution system functions; semantics.

1. Introduction

Manufacturing systems' enterprises seek for the employment of the latest technologies for elevating the features of the end products and services. This is a need in order to meet the incessant demand of products that in turn, creates an intensive competition in the development of highly efficient manufacturing systems. Currently, many research works are targeting the enhancement of the Manufacturing Execution Systems (MES) with novel ideas, as discussed in (Papakostas et al. 2012; Helo et al. 2014; Alexopoulos et al. 2016; Cheng et al. 1999). Fundamentally, MES binds the upper level (i.e., Enterprise Resource Planning (ERP)) with the lower level (i.e., factory shop floor) of manufacturing enterprises. Through this strategy, different organizations define and categorize the functionalities of MES into the so-called MES functions, which, in turn, are standardized by different organisations, such as the Manufacturing Enterprise Solutions Association (MESA), the International Society of Automation (ISA) or the Verein Deutscher Ingenieure (VDI) according to (Iarovyi et al. 2016). These functions are the source of the bond that MES provides for the manufacturing system.

Recently, the Embedded Systems Service-based Control for Open Manufacturing and Process Automation European project, known as eScop[1] project, has produced a research work in the scope of MES (Iarovyi et al. 2016). Particularly, the aforementioned article presents one of the main results of the eScop project, which is a Cyber-Physical System (CPS) presented as a flexible and reconfigurable solution for contemporary manufacturing systems. The eScop solution has defined a specific concept named as the Open Knowledge-Driven Manufacturing Execution System (OKD-MES) and created a framework for its implementation (Fumagalli et al. 2014; Negri et al. 2015). Principally, the OKD-MES allows enhancing the efficiency of factories by reducing the time and effort consumption needed for configuration, maintenance and scheduling orders and resources.

In this context, this article presents a compatible solution with the existing OKD-MES framework, which eScop project provided, for extending the implementation of MES functions. In other words, the presented solution is an extension for the existing framework in order to cover additional functionalities. More precisely, the contribution of this research work is a solution that enhances the flexibility, re-configurability and interoperability of event-driven manufacturing systems. This is done by the definition of standard MES functions and their encapsulation in services for permitting a modularized

[1] www.escop-project.eu

methodology to add new functionalities to the OKD-MES framework. Nevertheless, the presented solution is not restricted to systems that implement the OKD-MES concept.

The rest of the paper is structured as follows: Section 2 describes the research background. Then, Section 3 presents the architecture and implementation of the MES functions platform including its main components, their interactions and, as well, a case study. Afterwards, Section 4 provides the benefits and drawbacks of the presented approach. Finally, Section 5 concludes the article and presents the work to be done in the future.

2. State of The Art

2.1. Manufacturing Execution System

In the field of manufacturing systems, MES is considered as key supporting tools for production management because it plays the important role of bridging the ERP systems and the systems that operate at the physical field. In many companies, such bridge is delegated to a human activity, which compromises the optimization of the whole system. As an example, optimization might be applied on resources scheduling or labours exploitation. Moreover, such manual interaction often causes large efforts spent in the optimization of activities at ERP level, as well as implementation of important automation solutions at the shop floor level that will be further not used.

MES fills this gap in the automation chain, serving as a mediator or central layer integrating higher and lower levels of the ISA-95[2] automation pyramid. Nevertheless, MES cannot be seen as a simple and unique tool that transports information. Instead of that, MES represents a complex system where many functionalities are encompassed.

[2] https://www.isa.org/isa95/

The Manufacturing Enterprise Solutions Association[3] (MESA) defined 11 MES functions that are listed in Table 1 and described in more detail in (Iarovyi et al. 2016). MESA created the list in the late 90s and updated it along following decades as a union of all the functionalities that MES may have. However, the factory does not have to implement all the 11 functions to be optimally functional. In fact, the implementation of MES functions depends on several factors as e.g., the industry type or the automation level in the factory.

[Table 1 near here]

As highlighted in the Introduction section, different definition of MES functions may be found e.g., ISA95 or VDI[4]. MESA definition separates the MES functions in terms of scope of functionality. Hence, the eScop project involved MESA definition as the MES functions baseline.

On the other hand, from a research perspective, it is important to keep a comprehensive view embracing any possible function. In fact, it is recommended for any of the possible new research architecture to support the MES functions in order to keep the highest possible level of applicability in industrial environments. Different researchers have discussed MES application and MES functions (Marik and McFarlane 2005; Shaohong and Qingjin 2007). In fact, in (Younus et al. 2010), the authors review the development of MES applications and highlight the benefits of exploiting other technology fields (i.e. Information and Computer Technology). In the context of MES, the authors of (Valilai and Houshmand 2013) present an approach for MES as distributed systems that exploits service-oriented approach and multi-agent systems following the

[3] http://www.mesa.org/en/index.asp

[4] http://www.vdi.de

ISO 10303 standard. Besides, (Efthymiou et al. 2015) presents an approach for using knowledge-based systems for planning and designing manufacturing systems. The approach benefits from the features of Knowledge-Based System, which include reasoning and flexibility, and virtual factories concept to provide assessment during the design phase. Commonly, recent research work tends to fulfil the following requirements:

(1) The capability to have flexible functions that can guarantee complete automation of the proposed solution avoiding human activity by-pass;
(2) The capability to involve the operators, thus fostering the concept of man-in-the-loop, enabling the functions to be implemented by operators that can communicate with the MES systems thank to PDA (Personal Digital Assistant), RFID (Radio-Frequency Identification) reader, Barcode scanner, tablets, etc.
(3) The interoperability of the MES functions that can be guaranteed only if all the functions are implemented within the same software package or if a flexible alternative is available. This gap is the one fulfilled within the eScop project approach as presented in (Iarovyi et al. 2016).

In the domain of MES, several solutions can be found in the market nowadays. These solutions may vary from standalone solutions to service based solution or specific functionality oriented to general purpose oriented. In this research work, some of the solutions, which include POMSNet from Honeywell Process Solutions (“Life Sciences MES | by POMS” n.d.), ABB MES (“ABB Manufacturing Execution System - MES for Industrial Plants” n.d.), Rockwell Automation MES (“Manufacturing Execution Systems” n.d.), Siemens SIMATIC IT (“Industrial Automation Systems SIMATIC” n.d.) and GE Intelligent Platforms (“Plant Applications” 2016), have been studied in order to create a reference point for this research. More information is presented in Table 2.

[Table2 near here]

2.2.Interoperability

2.2.1. Communication

Communication is usually defined as the action of exchanging information. In the field of information technology, the communication is seen as a protocol for the exchange of information. The increased demand of communication protocols led to founding the World Wide Web Consortium (W3C) by Tim Berners-Lee at MIT in 1994 (Berners-Lee 1996). At that time, the W3C presented the Hypertext Transfer Protocol (HTTP) protocol. Since then, the W3C has provided various communication protocols for different use. On the other hand, in the same era, Organization for the Advancement of Structured Information Standards (OASIS) provided a reference architecture for managing the web services, such as the so called Service Oriented Architecture (SOA) (Papazoglou and Van Den Heuvel 2007; Jammes and Smit 2005). Accordingly, it was considered as an advantage for industrial devices to support the SOA architecture. Moreover, the SOA is implemented within the Devices Profile for Web Services (DPWS) for web services discovery and management (Jammes et al. 2005). The DPWS is based on Web Service Definition Language (WSDL). In parallel, W3C provided REpresentation State Transfer (REST) definitions for the web services. REST is based on HTTP request/response method. The request for REST services can be seen as the Create, Read, Update and Delete (CRUD) term. REST represents web resources using a uniform set of stateless operations. Furthermore, REST architecture supports several different data formats as e.g., XML, HTML, plain text and JSON. Although REST is not standardized yet, it features light weight processes and fault-tolerant (Ondřej Severa and Roman Pišl, 2015).

2.2.2. Semantics and Knowledge-Based Systems

Knowledge Representation (KR) and reasoning is a branch of Artificial Intelligence that describes and analyses human reasoning behaviour to support formal calculation and deduction. It defines symbols and languages that allow formalising knowledge in a precise semantics (Davis, Shrobe, and Szolovits 1993). In other words, KR allows the definition of the logical consequences that are understandable and automatically derivable by computer systems with reasoning algorithms (Motik and Rosati 2008). A key point in the knowledge formalization is the choice of the formal language, which must be sufficiently expressive to allow the description of the domain of interest and efficient enough for (1) not requiring too long reasoning time and (2) ensuring decidability (Negri et al. 2016). The container of KR formal descriptions is usually called a Knowledge Base (KB), which stores complex structured and unstructured information through a finite set of propositions on the domain of interest written in the chosen language. Knowledge-Based System (KBS) include both the syntax of the domain of interest (i.e., definition of rules to define acceptable interpretations of propositions) and a set of operators that provide a meaning or a value to the propositions (Yue, Liu, and Hunter 2007). On the other hand, KBSs offer a distributed data structure, contrarily to databases that provide a fixed data one (Astrova, Korda, and Kalja 2007). They answer to different data needs and it is the main reason for the origination of knowledge bases as an alternative to the traditional hierarchical and relational databases: the KR should not follow a tabular structure with rows and columns, but it is more convenient to use object modelling with a hierarchy of classes, subclasses, relationships and instances. As described in (Opdahl, Henderson-Sellers, and Barbier 2001), these features are perfectly provided by ontologies.

In literature, ontologies are defined as explicit specifications of a conceptualisation (Gruber 1995) for a shared understanding of information (Guarino 1998). In particular, a domain ontology is an abstract representation of reality within a certain scope. Ontologies are the natural candidates for implementing KBS, because they formalize knowledge about a domain improving expertise reusability in knowledge based systems (Giovannini et al. 2012). By their nature, ontologies do not have a specific application domain, but they may be the means to represent knowledge in any domain, in order to make it shared, explicit and formal. In particular, in the manufacturing domain, ontologies have a high potential application for unambiguous communication, to create a shared terminology and semantic alignment, meta-data in computational form for the information infrastructure (Schlenoff et al. 1999). The uses of ontology in the manufacturing domain could be several: from the knowledge sharing and reuse, to supporting interoperability in different systems.

The advent of modern smart technologies and distributed control in manufacturing environments has brought to light a promising application of ontologies. In fact, a production system consists of different independent and smart modules that are aware of the capabilities that they can offer to the system but do not have any knowledge on the role they have to play together in the production from a systemic point of view (M. Garetti et al. 2013; Marco Garetti, Fumagalli, and Negri 2015). Ontologies are the perfect means for providing such kind of knowledge to distributed control architectures. In fact, in centralized control architectures, the different system components do not require control information on the role. Instead of that, the logics of the system has been integrated in the design of the system. On the contrary, distributed control is made of smart components that act as independent elementary modules that perform their local control and require a centralized representation of their role in the system. This role must be formalized through a shared representation of the domain, characteristics that are supplied by ad hoc ontologies (M. Garetti and Fumagalli 2012). The modular approach of this kind of production systems allows reducing building, ramp up and reconfiguration time of manufacturing automation systems significantly, because when a module is removed, modified or included, the knowledge of the system - on which the control is based - is easily updated.

The available semantic languages that can be used for implementing ontologies are several, each of them characterized by different syntax, reasoning capabilities and complexity among other features (Giunchiglia et al. 2010). A comparison of semantic languages against a set of collected requirements from the manufacturing domain is found in (Negri et al. 2016). The languages that are advised for this domain are the Web Ontology Language[5] (OWL) and the OWL sublanguages (Lite and DL). They are based on Resource Description Framework[6] (RDF) and are able to represent semantic information in a simple and meaningful manner (through the so-called Triples: Subject-Predicate-Object). They can be queried with the SPARQL language (i.e., recursive acronym for SPARQL Protocol and RDF Query Language) to retrieve the information stored in the KBS (Negri et al. 2016).

2.3. Open Knowledge-Driven Manufacturing Execution Systems

Referring to the previous subsections, a particular approach for the implementation of the MES was proposed. This approach aims to provide high level of modularity and re-configurability for the MES solution and underlying automation field layers. The approach combines the capabilities of embedded devices and web services with the advanced approach for the knowledge management and will be referred further as OKD-MES.

[5] http://www.w3.org/TR/owl-ref

[6] http://www.w3.org/RDF/

The OKD-MES provides a reference architecture including four core layers: Physical Layer (PHL), Representation Layer (RPL), Orchestration Layer (ORL), Visualization Layer (VIS), and a layer of loosely-coupled deployable MES functions as can be observed on Figure 1.

[Figure 1 near here]

The PHL addresses the problem of integration of factory shop-floor equipment to the OKD-MES ecosystem. PHL is deployed in the form of RESTful web-service enabled controllers. The PHL devices expose the contract and related services required to interact with the controlled equipment. As well, the part of discovery mechanism is implemented in the PHL, which allows keeping the system representation in sync with the real world. The RPL hosts a manufacturing domain ontology that serves as semantic representation of the production systems, that is instantiated for the specific industrial application case. As such, it offers the capabilities of ontologies, which is presented in Section 3.2 allowing interoperability and knowledge sharing along Cyber-Physical Systems in the OKD-MES (Legat et al. 2014). Having a knowledge base in its core, the RPL is capable for some complex data querying and possibly reasoning. Considering the implementation of independent service and capabilities description, the RPL allows an interface for a dynamic dependency injection. In addition to the ontology, the RPL implements part of discovery mechanism, which interacts with PHL devices, through the ontology service that exposes the ontology as a service on the web service architecture.

Meanwhile, orchestrators in the ORL resolve the coordination task in OKD-MES solution. The orchestrators are capable to execute service compositions based on abstract process definitions. Such definitions can be resolved in interactions with RPL to executable service invocations, which are handled by ORL. The process definitions are also defining the error and emergency handling in OKD-MES.

The Visualization Layer, on the other hand, provides a generic interface between the users and the application. Each OKD-MES component can use declarative User Interface (UI) definition, which is materialized to a web application interface by the VIS. This declarative definition allows a dynamic UI generation based on user specified rules.

The four layers mentioned above provide the ground for implementation of MES functions. Technically, any MES function can be defined as a process definition with a corresponding UI. Unfortunately, such approach will lead to an overcomplicated process definition and will reduce systems performance due to the natural compromises between the flexibility and performance.

3. Manufacturing Execution Systems Platform Architecture

Lately, with the available computational resources, the installation of manufacturing system, configuration and running costs and time have been significantly reduced (Cheng et al. 1999). Thanks to the intensive research work that has been conducted on this field. Similarly, this research work presents a dynamic, flexible and reconfigurable platform for providing the MES functionalities for manufacturing systems as services. The platform employs the KBS and RESTful web services to allow dynamic and autonomous interaction of the components in the manufacturing system. With such solution, it is expected for the manufacturing systems to be used as a distributed solution or cloud-based application.

The suggested platform allows the user to implement particular MES functions. According to the user needs, MES functions may be provided as web services. Accordingly, these services might require some logical or arithmetical calculations. While the calculations might be defined in the form of the functional scripts. This section presents the structure of the platform and defines the workflow of the platform. It consists of four subsections; in the first one, the components of the platform are described. The second subsection presents the ontology model that has been used for building a knowledge-based system. Meanwhile, the third subsection shows the interactions with the provided solution. Finally, the forth subsection provides a case scenario as a proof of the concept.

3.1.Components

As the majority of web applications, this MES functions platform provides its own services through the web. As well, it is designed to be configurable in running time manners. Accordingly, the platform needs to have an interface with the external environment and systems. Moreover, it has to possess some capabilities to persist internal information. Besides, the platform should provide the consumers (the manufacturing systems) with a specific functionality. To satisfy the aforementioned requirements, the presented approach consists of a Service Manager (SM), which facilitates the interactions with the environment, an Ontology Manager (OM) for providing the tools to persist the internal information and a Function Manager (FM) that enables the processing of the information in the system and supports the binding to the corresponding interfaces. Some information about the component design is presented in the following subsections, and implementation details can be found in KPI case study subsection.

[Figure 2 near here]

3.1.1. Service Manager

The SM provides the platform with a proper interface for the communications. The SM contains two main components; the RESTful interface and the Services Composition and Decomposition Unit (SCDU). The RESTful interface supplies the SM with the Application Program Interface (API) capabilities to provide and consume the RESTful services. Meanwhile, the SCDU decomposes the received RESTful requests or the responses for certain requests. Similarly, it composes the responses for the requested services or for the requests, which the platform requires. As a result, the SCDU binds the platform with the RESTful interface by providing a translation for the incoming/outgoing messages. As shown in Figure 2 , the SM transforms the incoming requests or returned responses into Java Script Object Notation (JSON) and then passes it to the FM for further actions, and vice versa, the SM transforms the objects defined by the FM that are transmitted through the RESTful interface.

3.1.2. Ontology Manager

Re-configurability and flexibility are considered as the main features of the proposed approach. Thus, the MES platform employs the KBS technology. As many KBS based on ontologies, the current approach requires the model to be defined. After that, the user can populate the model with information relevant for the application case. Within this approach, the ontology is managed by a specific component – the OM. The OM provides the platform with the proper information via querying and updating the information model. To achieve that, the OM enables SPARQL-based services for providing the required functionality.

3.1.3. Function Manager

Finally, the third main component in the platform is the Function Manager (FM). The FM provides the platform with logic processing module. As well, the FM contains the functional scripts, which are used to provide MES its functionality. In this context, the user defines the functional scripts that fits the manufacturing needs, and then, inject them in the FM. Besides, the FM binds the other managers (i.e., SM and OM). With this architecture, the FM contains the core of the MES functions platform, which is called

Processing Unit (PU). The PU provides the runtime environment for the functional scripts.

3.2.Ontology Model

The ontology model supplies the platform with the required information, i.e. configurations, services and functional scripts accessibility. As presented in the previous subsection, the ontology model is managed through the OM. Figure 3 shows the ontology model for the platform. It contains the following main classes: *OkdMesLayer*, *Configurations*, *MesFunction*, *Service*, *FunctionalScript* and *Parameter*.

[Figure 3 near here]

The *OkdMesLayer* class holds the information about the accessibility of the platform. This class includes four datatype properties; *id*, *name*, *host* and *port*. *OkdMesLayer* is linked to *Configurations*, *Service* and *MesFunction classes* using *needsConfig*, *hasService* and *hasMesFunction* object properties respectively. The *Configurations* class allows the *OkdMesLayer* class to hold configuration parameters regarding the other components in the manufacturing system. It contains *deviceCatalogueUrl* datatype property for exploring the manufacturing system services. *phlEvents* datatype property for providing a list of events, which the platform should subscribe. Finally, *eventListenerUrl* datatype property for determining the Unique Resource Locator (URL) where the platform will receive notifications of the PHL events. Then, the second link connects the layer with the web services. This means that the platform might have some service instances, which are not related to the MES functions.

Thirdly, the *MesFunction* class includes two datatype properties: *id* and *name*. The *MesFunction* class is linked to *Service* and/or *FunctionalScript* via *hasService* and *hasFunctionalScript* object properties**,** respectively. This means that the MES function could have background functions that run without a request from a service. Opposite to that, it can serve certain services without having a background functions. Then, the *Service* class contains eight datatype properties; *id*, *url*, *method*, *reqBody*, *reqQuery*, *reqPram*, *responseStatus* and *responseBody*. As a RESTful service, the *url* and the *method* are used for routing and validating the correctness of the request. *reqBody*, *reqParam* and r*eqQuery* hold the request information. Meanwhile, the *resStatus* and *resBody* represent the response for the requested service. In this context, the *resStatus* defines the response http status code. It should be noted that the value of the *resStatus* plays a role in the response of the received requests. More illustration is presented in the next subsection. The *Service* class is connected to the *FunctionalScript* class by *hasFunctionalScript* object property. The *FunctionalScript* class represents a logical and/or arithmetic set of operation, which might be called by a service invocation or as a background function. The *FunctionalScript* class includes three datatype properties; *id*, *name* and *url*. The *name* is used for calling the function while the *url* presents the accessibility for the function script. In this way, the function script can be a cloud resource, which is requested once it is needed. Then the *FunctionalScript* class is linked to *Parameter* class using *hasParameter* property. The Parameter class consists of *name*, *type* and *value* datatype properties. This class is used for two reasons:

(1) Passing parameters to the functional scripts.

(2) Storing variables in the platform where functional scripts can share data.

3.3.Interactions

The interactions of such an approach can be seen from two different points of views. Firstly, how the user will setup and run the platform. Secondly, how the platform will provide functionality to the manufacturing system. In this subsection, an illustration is provided for demonstrating the two integration scenarios.

3.3.1. User Interactions

Alike of any application, the MES platform requires a setting up before the user can run it in the manufacturing system. Therefore, an elucidation of the activities that the user should conduct for running the platform is presented in Figure 4. The user starts by applying a study of the feasibility of using MES functions platform in the manufacturing system. This feasibility study covers the need of the MES function in the manufacturing system. As an example, in manual production line, the most feasible function for the user could be Labour Management. Once it is feasible, the user is required to populate the ontology model with proper instances. In this stage, the user is entitled for defining the web services that are required for the platform to serve. Moreover, the user determines the functional scripts that the platform requires. Afterwards, the user uploads the instances to the platform.

[Figure 4 near here]

Thereupon, the user examines the existence of the functional scripts. For instance, these functional scripts could be algorithms for optimization or Key Performance Indicators (KPI) formulas. Consequently, the user updates the platform. Such update might only consist of URLs for these functional scripts in the ontology instances or code scripts that need to be uploaded to the platform. Finally, the user runs the platform since it is populated with ontology instances and functional scripts.

3.3.2. Manufacturing System Interactions

The second integration scenario addresses the runtime flow work of the platform and its interactions with the manufacturing system. Once the platform is set on running mode, it subscribes to all events in the *Configurations* class in the ontology model.

The subscription of PHL events are an option of the user design to handle PHL information. An example can be seen in Resource Status and Allocation MES function.

The status of PHL resources is propagated through events notification. The sequence of the subscription of the PHL events is shown in Figure 5.

[Figure 5 near here]

The subscription starts with reading the configurations of the platform. Afterwards, the FM requests the SM to perform subscription services for each *eventID* in the configuration of the platform.

After the subscription process, the production phase starts. Figure 7 presents a request-response life cycle in the platform. An application, such as an orchestrator sends a request message to the platform. The request message reaches the SM first through the interface. Consequently, the SM sends a notification to the FM informing that a new request message has been received. Figure 6 presents the notification body of the service request. The PU, which is the core of the FM, generates a SPARQL query to validate service.

[Figure 6 near here]

The validation of the services is combined within the KBS. As highlighted in the previous subsection, the user defines all the services, which the platform will serve. Therefore, the service validation can be extracted from response of the SPARQL query.

[Figure 7 near here]

Figure 8 shows the SPARQL query, which is used for validating the request. As shown, the SPARQL query returns the service, functional script, response status and response body of the provided service URL and service method. The result of the query could return an empty result in case the URL and method are not matching. The empty result leads to "*not found"* response. On the other hand, the non-empty result means that the requested service is registered in the platform. Depending on the *resResult* datatype property, the platform takes its action. For instance, if the *resStatus* datatype equals "0", then the PU responds to the requested service with the result of the functional script that the ontology manager provided. Otherwise, the response is directly provided by the result of the *resBody* of the ontology manager's results. An example of the implementation is illustrated in the next subsection.

[Figure 8 near here]

3.4.KPI case study

In order to demonstrate the usage of the platform, an example of deploying the *Performance Analysis* MES function is illustrated in this subsection. This use case demonstrates the actual FASTory[7] assembly line located in Tampere University of technology in Tampere, Finland. The FASTory line contains 12 workstations equipped with IoT devices following the eScop project approach which described in (Ondřej Severa and Roman Pišl, 2015). These workstations contain robot resource and conveyor resource that are represented by IoT devices as discussed in (Mohammed et al. 2016). In total, 24 IoT devices publish events that describe the changes in the status as depicted in Figure 9.

[Figure 9 near here]

Regarding this scenario, the performance analysis MES function performs the following activities:

(1) It measures continuously the exploitation ratio of a resource in the PHL as presented in equation (2).
(2) It serves a GET method for retrieving the utilization efficiency KPI that is defined in ISO 22400-2:2014 standard (International Organization for Standardization 2014).

[7] http://www.tut.fi/en/fast/fastory/index.htm

Following the definition of the KPI definition in the aforementioned standard, equation 1 results with the busy time for a resource. Following that, equation 2 results the exploitation KPI by calculating the ration between total running time with respect to the total busy time.

[Equation 1 near here]

[Equation 2 near here]

Where,

$T_{Idle \to busy \to Idle}$: the period that the resource has been in the busy state.

T_{busy}: the summation of the periods, which the resource has been in busy state.

T_{total}: the total period since the system is running.

[Figure 10 near here]

As appears in Figure 10 , the received event from the PHL follows the eScop templates for event. The id and *resourceId* represent the event ID and the event publisher ID respectively. The *lastEmitted* provides information about the last emitting of the event. The format of the time in milliseconds since 00:00:00.00 01/01/1970. The specific information of the event is held in the payload. In this manner, the "v" key represents the value of the resource stat, the "q" key represents the quality of the value and "t" key provides the current time once the event is issued.

To do so, the MES functions should subscribe to the resource state change event in the PHL. In this context, the platform requests from the device catalogue (*deviceCatalogue* datatype for *Configurations* class in Figure 11) the specific URL of the subscription. The platform subscribes in PHL by providing the notification URL for the event (*eventListenerUrl* datatype of Configurations class in Figure 11).

[Figure 11 near here]

With reference to this, the platform performs the functional script, which is shown in Figure 12. Each time the platform receives an event with *stateChanged* id and the status is *IDLE*. The functional script performs the arithmetic equations (1) and (2). The *OM.setParameter* and *OM.getParameter* are embedded functions for manipulating parameters in KBS. It has to be noted that the functional scripts are in JavaScript language because the platform has been built using *Node.js*[8].

[Figure 12 near here]

Then the platform responds with the KPI value once the GET KPI service is invoked. The service uses *calculateKPI* functional script (see Figure 13).

[Figure 13 near here]

4. Discussion

The employment of the proposed framework allows the dynamic system reconfiguration as the functional requirements and corresponding components are being bound in the runtime. The dependency injection mechanism outlined in the OKD-MES concept was improved and implemented in the presented platform. In combination with the ontology based configuration and communication in the system, the platform enables "cooperation without coordination". Furthermore, the community driven evolution of the platform functionality is enabled. Considering that the design of the system is based on the open and widely accepted web standards, is technology agnostic and is enabling the proper isolation of abstraction levels. Some of the technical and social challenges of the platform acceptance are addressed and possible developer community can be broadened. The modularity of the system encompasses the small steps migration from the

[8] https://nodejs.org/en/

conventional MES solutions to OKD-MES and possible further improvement of the systems.

As it can be compared with ready-made solutions, the presented platform depends on the loosely coupled concept of web services while some of the listed solutions in Table 2 depends on the stand-alone installation concept with access to the web for extra features. This may decrease the installation cost and time and more important, it eliminate the dependency of the available resources. On the opposite, the presented platform lack of maturity and requires more development to address the challenges as sellable product. In regards of functionality, the presented MES platform allows the user to define the logic of the functionality according to the manufacturing systems needs while the ready-made solutions provide more rigid functionalities.

One of the most important disadvantages is the increased demand in computational resources to maintain the performance of the system. As any loosely-coupled system, current, OKD-MES platform has communication and configuration overhead, comparing to the tightly-coupled analogues. Furthermore, the late binding based on the ontology increases its complexity, making the process more resource demanding. Another challenge is the security of the system. The use of widely accepted web standards and internet-based communication leads to an increased amount of vulnerability points in the system. Besides the technical security, there is a wider challenge to overcome the “digital angst” – overall scepticism towards the web, which is especially strong in the established domains such as manufacturing. Finally, to exploit all the benefits of the proposed platform there is a need to modify the paradigm in the development of surrounding systems. For example, the controllers in the manufacturing lines should expose more metadata about themselves, and provide the functionality of the higher level of abstraction.

The authors claim that the advantages of the proposed approach are addressing emerging needs in growing factory information systems, while some of the drawbacks and challenges are showing a trend to be resolved by the advance of technology and overall digitalization in all domains of human life.

5. Conclusion

The article has described an architecture for implementing MES systems. The presented architecture is expected to serve the manufacturing systems through MES functionalities. The platform showed an easy configurability and flexibility in terms of setting up by the user. Moreover, it addressed the genericity of serving different manufacturing systems. Since the platform relies on *Node.js*, the installation is expected to be simple and fast. This platform has been tested on a discrete assembly line in Tampere University of Technology where the target was to highlight the functionality of the platform.

Future work should address development of relevant business models to support OKD-MES principles as migration towards knowledge-driven approaches would require rethinking established industrial practices. The change would need the actions on all the levels of OKD-MES architecture, starting from the controller devices in charge of industrial enjoinment to the higher-level information systems, where the shift from databases to knowledge bases should be performed. This would require development of new methodologies to include proved methods and powerful and easy-to-use tools. However, the authors believe that the development of such tools and methods will be growing and supported due to broad availability of developers and experts working with the web standards, which are also in the core of the presented architecture. In addition, well-established tools are available from the general software engineering discipline. Those tools can be easily adopted in the field of industrial automation. Moreover, the platform is planned to be developed and tested in terms of reaching a productive tool which allows distribution and exploitation in other research work.

References

"ABB Manufacturing Execution System - MES for Industrial Plants." n.d. Accessed October 4, 2017. http://new.abb.com/cpm/manufacturing-execution-system-mes-mom.

Alexopoulos, Kosmas, Sotiris Makris, Vangelis Xanthakis, Konstantinos Sipsas, and George Chryssolouris. 2016. "A Concept for Context-Aware Computing in Manufacturing: The White Goods Case." *International Journal of Computer Integrated Manufacturing* 29 (8):839–49. https://doi.org/10.1080/0951192X.2015.1130257.

Astrova, Irina, Nahum Korda, and Ahto Kalja. 2007. "Storing OWL Ontologies in SQL Relational Databases." *International Journal of Electrical, Computer and Systems Engineering* 1.

Berners-Lee, Tim. 1996. "WWW: Past, Present, and Future." *Computer* 29 (10):69–77.

Cheng, Fan-Tien, Eric Shen, Jun-Yan Deng, and Kevin Nguyen. 1999. "Development of a System Framework for the Computer-Integrated Manufacturing Execution System: A Distributed Object-Oriented Approach." *International Journal of Computer Integrated Manufacturing* 12 (5):384–402. https://doi.org/10.1080/095119299130137.

Davis, Randall, Howard Shrobe, and Peter Szolovits. 1993. "What Is a Knowledge Representation?" *AI Magazine* 14 (1):17. https://doi.org/10.1609/aimag.v14i1.1029.

Efthymiou, Konstantinos, Konstantinos Sipsas, Dimitris Mourtzis, and George Chryssolouris. 2015. "On Knowledge Reuse for Manufacturing Systems Design and Planning: A Semantic Technology Approach." *CIRP Journal of Manufacturing Science and Technology* 8 (Supplement C):1–11. https://doi.org/10.1016/j.cirpj.2014.10.006.

Fumagalli, Luca, Simone Pala, Marco Garetti, and Elisa Negri. 2014. "Ontology-Based Modeling of Manufacturing and Logistics Systems for a New MES Architecture." In *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World*, 192–200. IFIP Advances in Information and Communication Technology. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44739-0_24.

Garetti, M., and L. Fumagalli. 2012. "P-PSO Ontology for Manufacturing Systems." *IFAC Proceedings Volumes*, 14th IFAC Symposium on Information Control Problems in Manufacturing, 45 (6):449–56. https://doi.org/10.3182/20120523-3-RO-2023.00222.

Garetti, M., L. Fumagalli, A. Lobov, and J. L. Martinez Lastra. 2013. "Open Automation of Manufacturing Systems through Integration of Ontology and Web Services." *IFAC Proceedings Volumes*, 7th IFAC Conference on Manufacturing Modelling, Management, and Control, 46 (9):198–203. https://doi.org/10.3182/20130619-3-RU-3018.00169.

Garetti, Marco, Luca Fumagalli, and Elisa Negri. 2015. "Role of Ontologies for CPS Implementation in Manufacturing." *Management and Production Engineering Review* 6 (4):26–32. https://doi.org/10.1515/mper-2015-0033.

Giovannini, Antonio, Alexis Aubry, Hervé Panetto, Michele Dassisti, and Hind El Haouzi. 2012. “Ontology-Based System for Supporting Manufacturing Sustainability.” *Annual Reviews in Control* 36 (2):309–17. https://doi.org/10.1016/j.arcontrol.2012.09.012.

Giunchiglia, Fausto, Feroz Farazi, Letizia Tanca, and Roberto De Virgilio. 2010. “The Semantic Web Languages.” In *Semantic Web Information Management*, 25–38. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04329-1_3.

Gruber, Thomas R. 1995. “Toward Principles for the Design of Ontologies Used for Knowledge Sharing?” *International Journal of Human-Computer Studies* 43 (5):907–28. https://doi.org/10.1006/ijhc.1995.1081.

Guarino, Nicola. 1998. “Formal Ontology and Information Systems.” In *Proceedings of FOIS’98*, 3–15. Trento, Italy: IOS Press.

Helo, Petri, Mikko Suorsa, Yuqiuge Hao, and Pornthep Anussornnitisarn. 2014. “Toward a Cloud-Based Manufacturing Execution System for Distributed Manufacturing.” *Computers in Industry* 65 (4):646–56. https://doi.org/10.1016/j.compind.2014.01.015.

Iarovyi, S., W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra. 2016. “Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems.” *Proceedings of the IEEE* 104 (5):1142–54. https://doi.org/10.1109/JPROC.2015.2509498.

“Industrial Automation Systems SIMATIC.” n.d. Newton_ps-access. Accessed October 4, 2017. https://www.siemens.com/global/en/home/products/automation/systems/industrial.html.

International Organization for Standardization, (ISO). 2014. “Automation Systems and Integration -- Key Performance Indicators (KPIs) for Manufacturing Operations Management.” https://www.iso.org/standard/56847.html.

Jammes, F., H. Smit, J. L. M. Lastra, and I. M. Delamer. 2005. “Orchestration of Service-Oriented Manufacturing Processes.” In *2005 IEEE Conference on Emerging Technologies and Factory Automation*, 1:8 pp.-624. https://doi.org/10.1109/ETFA.2005.1612580.

Jammes, F, and Harm Smit. 2005. “Service-Oriented Paradigms in Industrial Automation.” *Industrial Informatics, IEEE Transactions On* 1 (1):62–70.

Legat, Christoph, Christian Seitz, Steffen Lamparter, and Stefan Feldmann. 2014. “Semantics to the Shop Floor: Towards Ontology Modularization and Reuse in the Automation Domain.” In *Proceedings of the 19th IFAC World Congress*, 3444–49.

“Life Sciences MES | by POMS.” n.d. Accessed October 4, 2017. http://www.poms.com/.

“Manufacturing Execution Systems.” n.d. RockwellAutomation.Com. Accessed October 4, 2017. http://www.rockwellautomation.com/global/products/manufacturing-execution-systems/overview.page?pagetitle=Manufacturing-Execution-Systems&docid=6309f28d5f4a5287357c60080fa16d44.

Marik, V., and D. McFarlane. 2005. “Industrial Adoption of Agent-Based Technologies.” *IEEE Intelligent Systems* 20 (1):27–35. https://doi.org/10.1109/MIS.2005.11.

MESA International. 2011. “MESA White Paper #39: MESA Model Evolution.” https://services.mesa.org/ResourceLibrary/ShowResource/73b23d9e-133e-456b-b844-d7ba5ff8278a.

Mohammed, W. M., A. Lobov, B. R. Ferrer, S. Iarovyi, and J. L. M. Lastra. 2016. “A Web-Based Simulator for a Discrete Manufacturing System.” In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 6583–89. https://doi.org/10.1109/IECON.2016.7793563.

Motik, Boris, and Riccardo Rosati. 2008. “Reconciling Description Logics and Rules.” *J. ACM* 57 (5):30:1–30:62. https://doi.org/10.1145/1754399.1754403.

Negri, Elisa, Luca Fumagalli, Marco Garetti, and Letizia Tanca. 2016. “Requirements and Languages for the Semantic Representation of Manufacturing Systems.” *Computers in Industry*, Emerging ICT concepts for smart, safe and sustainable industrial systems, 81 (Supplement C):55–66. https://doi.org/10.1016/j.compind.2015.10.009.

Negri, Elisa, Luca Fumagalli, Marco Macchi, and Marco Garetti. 2015. “Ontology for Service-Based Control of Production Systems.” In *Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth*, 484–92. IFIP Advances in Information and Communication Technology. Springer, Cham. https://doi.org/10.1007/978-3-319-22759-7_56.

Ondřej Severa, and Roman Pišl. n.d. “REST-Enabled Physical Devices in Service Oriented Architecture.” In *OPEN KNOWLEDGE-DRIVEN MANUFACTURING & LOGISTICS, THE ESCOP APPROACH*, edited by Stanisław Strzelczak, Pavel Balda, Marco Garetti, and Andrei Lobov, 341–54. Warsaw University of Technology Publishing House, Warsaw 2015.

Opdahl, A. L., B. Henderson-Sellers, and F. Barbier. 2001. “Ontological Analysis of Whole–part Relationships in OO-Models.” *Information and Software Technology* 43 (6):387–99. https://doi.org/10.1016/S0950-5849(00)00175-0.

Papakostas, N., D. Mourtzis, G. Michalos, S. Makris, and G. Chryssolouris. 2012. “An Agent-Based Methodology for Manufacturing Decision Making: A Textile Case Study.” *International Journal of Computer Integrated Manufacturing* 25 (6):509–26. https://doi.org/10.1080/0951192X.2011.637963.

Papazoglou, Mike P., and Willem Jan Van Den Heuvel. 2007. “Service Oriented Architectures: Approaches, Technologies and Research Issues.” *VLDB Journal* 16 (3):389–415. https://doi.org/10.1007/s00778-007-0044-3.

“Plant Applications.” 2016. GE Digital. May 19, 2016. https://www.ge.com/digital/products/plant-applications.

Schlenoff, Craig I., Robert W. Ivester, Don E. Libes, Peter O. Denno, and Simon Szykman. 1999. “An Analysis of Existing Ontological Systems for Applications in Manufacturing and Healthcare.” *NIST Interagency/Internal Report (NISTIR) - 6301*, January. https://www.nist.gov/publications/analysis-existing-ontological-systems-applications-manufacturing-and-healthcare.

Shaohong, J., and M. Qingjin. 2007. “Research on MES Architecture and Application for Cement Enterprises.” In *2007 IEEE International Conference on Control and Automation*, 1255–59. https://doi.org/10.1109/ICCA.2007.4376562.

Valilai, Omid Fatahi, and Mahmoud Houshmand. 2013. “A Collaborative and Integrated Platform to Support Distributed Manufacturing System Using a Service-Oriented Approach Based on Cloud Computing Paradigm.” *Robotics and Computer-Integrated Manufacturing* 29 (1):110–27. https://doi.org/10.1016/j.rcim.2012.07.009.

Younus, Muhammad, Cong Peiyong, Lu Hu, and Fan Yuqing. 2010. “MES Development and Significant Applications in Manufacturing -A Review.” In *2010 2nd International Conference on Education Technology and Computer*, 5:V5-97-V5-101. https://doi.org/10.1109/ICETC.2010.5530040.

Yue, Anbu, Weiru Liu, and Anthony Hunter. 2007. “Approaches to Constructing a Stratified Merged Knowledge Base.” In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 54–65. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-75256-1_8.

Tables

Table 1. MESA International MES Fonctions briefe description (MESA International 2011)

Function	**Description**
Resource allocation and status	Manages resources information, providing detailed history and status on real time
Operations/Detail Scheduling	Provides sequencing, recognizing alternative and overlapping/parallel operations
Dispatching Production Units	Manages flow of production units
Document Control	Controls records/forms to be maintained with the production unit
Data Collection/Acquisition	Obtains the intra-operational production and parametric data from the factory floor
Labour Management	Provides optimisation of the labour exploitation
Quality Management	Provides real time information to assure product quality
Process Management	Monitors production and provides or corrects decision support to improve process activities
Maintenance Management	Tracks maintenance activities and provides instance solutions
Product Tracking and Genealogy	Provides the status information of work activities. Also it may generate historical information for the products that have been produced
Performance Analysis	Presents the performance (i.e. KPIs) of the facility for more study and analysis.

Table 2. Ready MES solution in the market.

MES System	**Key Feature**	**Other information**
POMSnet by Honeywell Process Solutions	Specification Management, Material Management, Quality Management, Inventory Management, Production Management, Weigh and Dispense, Equipment Management, Process Control Integration and Standard Reports	Integration with process control, ERP, CAPA web-based, built on Microsoft´s .NET technology (For life science), complies with CFR 21 Part 11 and ISA manufacturing standards.
ABB MES	Warehouse management, Assess management (preventive maintenance) support for industry type (descrete, food, metals, mining), support by role (business management, operations, management, shopfloor operators, mainteinace supervisor, supply manager, IT manager)	ISA-95 based solution integration with ERP
Rockwell Automation MES	Supplier Management, Production Inventory, Analysis and reporting tool (operational content for identification and root cause analysis of KPI), Material Management	
Siemens SIMATIC IT	Modular architecture which provies solutions for some of the manufacturing systems functionalities via SIMATIC application family (Historian, Interspec, Line Monitoring Systems, Preactor Advanced Planning and Scheduling, Unicam, Unilab, eBR)	conforms ISA-95, interaction with product specification management, ERP, other business systems
GE Intelligent Platforms	Modular, consists e.g. Quality, Efficiency, Production and Bach Analysis applications , other Prophecy Suit	uses SOA approach and OPC UA standard,

	applications such as Data Management, Maintenance Gateway, Scheduler, Workflow, Enterprise, Trouble-shooter and Cause, Historian, Pulse, WebSpace/GlobalView, Portal	

Figures’ Captions

Equations

$$T_{busy} = \sum T_{Idle \rightarrow Busy \rightarrow Idle} \tag{1}$$

$$KPI_{exploitation} = \frac{T_{total}}{T_{busy}} \times 100\% \tag{2}$$